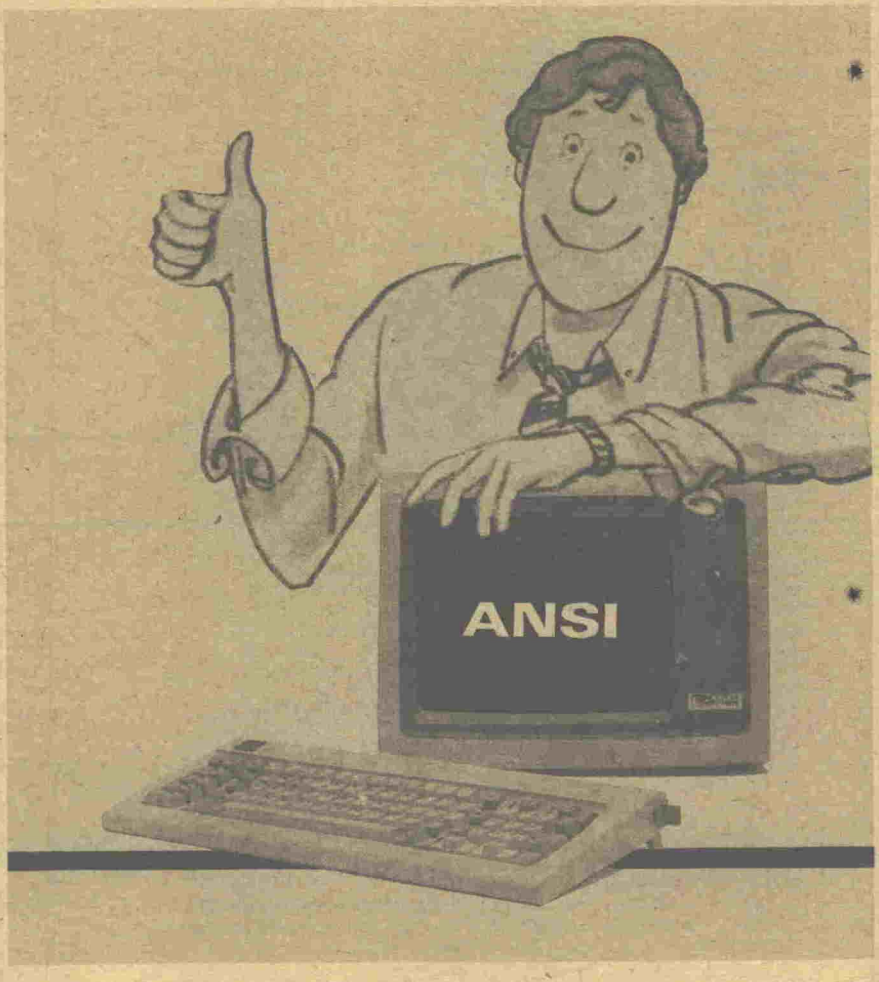
บุญแสง สุรวัฒนบุตร

สร้างยูทิลิตี้จากรหัส ANSI

รหัส ANSI ควบคุมการทำงานของจอภาพ ช่วยเปลี่ยนสีตัวอักษร ได้อย่างมาตรฐาน



ในแผ่นของดอสจะมีไฟล์อยู่ประเภทหนึ่ง ซึ่งให้ไว้สำหรับนำไป install ระบบการทำงานของอุปกรณ์เครื่องคอมพิวเตอร์ที่เรียกว่าโปรแกรมย่อยควบคุมอุปกรณ์ (device driver) ไฟล์ที่ชื่อว่า ANSI.SYS ก็เป็นไฟล์หนึ่งที่ใช้สำหรับการโหลดรหัสของ ANSI (ANSI escape sequence) เพื่ออนุญาตให้ใช้รหัส ANSI ในการควบคุมการทำงานของจอภาพ การเรียกใช้โปรแกรมย่อยควบคุมอุปกรณ์ต้องกำหนดลงในไฟล์ CONFIG.SYS ตัวอย่างแสดงในรูปที่ 1 ในบรรทัดที่ 3 ของไฟล์ CONFIG.SYS ได้กำหนดการใช้รหัสของ ANSI อย่าลืมว่าไฟล์ CONFIG.SYS จะถูกเรียกใช้เมื่อบูตเครื่องเท่านั้น ดังนั้นหากผู้อ่านทำการแก้ไขไฟล์ CONFIG.SYS แล้วจะต้องทำการบูตเครื่องใหม่

รหัส ANSI มีอยู่หลายคำสั่งด้วยกัน ก่อนที่จะเริ่มการสร้างยูทิลิตี้ในเทอร์โบปาสคาลขอให้ผู้อ่านทราบไว้ว่าถ้าหากต้องการใช้รหัส ANSI ในการควบคุมการทำงานของจอภาพจะไม่สามารถใช้ยูนิต Crt ของเทอร์โบปาสคาลได้ ถึงแม้ว่ายูนิต Crt ของเทอร์โบปาสคาลจะมีคำสั่งต่าง ๆ ที่มีประสิทธิภาพมากก็ตาม แต่รหัส ANSI มีความได้เปรียบในด้านความเป็นมาตรฐานมากกว่า ดังนั้นการทราบถึงรหัส ANSI ของระบบที่ใช้งานอยู่ ซึ่งมักจะมีค่าคล้ายกันในแต่ละระบบ จะสามารถทำให้ผู้อ่านสามารถสร้างยูทิลิตี้ที่ใช้รหัส ANSI ได้ โดยอาจใช้ปาสคาลมาตรฐานก็ได้ เนื่องจากการส่งค่าของรหัส

```
C:\>type config.sys
FILES=20
BUFFERS=15
DEVICE=C:\DOS\ANSI.SYS
```

รูปที่ 1 การกำหนดการใช้ ANSI.SYS

ANSI สามารถส่งได้ง่ายด้วยคำสั่ง Write เท่านั้น นอกจากนี้รหัสของ ANSI ยังมีคุณสมบัติพิเศษประการหนึ่งคือ ถ้าหากเราใช้คำสั่ง TextColor ของยูนิต Crt ในการเปลี่ยนสีของตัวอักษร แต่เมื่อจบโปรแกรมแล้วออกมาที่ดอส สีของตัวอักษรจะสิ้นสุดทันทีที่ใช้คำสั่ง cls ในดอส แต่สำหรับรหัสการเปลี่ยนสีตัวอักษรของ ANSI รวมทั้งรหัสอื่น ๆ ของ ANSI จะมีผลถึงการทำงานในดอสด้วย

เราจะเริ่มการสร้างยูทิลิตี้ชุดแรกจากรหัส ANSI ด้วยตารางในรูปที่ 2 แสดงรหัส ANSI ที่เกี่ยวข้องกับการทำงานของเคอร์เซอร์ รายละเอียดของรหัสต่าง ๆ แสดงไว้ในตาราง ลักษณะพิเศษของการส่งรหัส ANSI ประการหนึ่งคือ สามารถส่งรหัสเป็นตัวเลขหรือสตริงของตัวเลขนั้นก็ได้ เช่น จากรูปที่ 2 รหัสที่ใช้ในการเปลี่ยนตำแหน่งของเคอร์เซอร์ (CUP) สามารถใช้คำสั่งในเทอร์โบปาสคาลต่อไปนี้ ซึ่งมีความหมายเหมือนกัน คือเปลี่ยนตำแหน่งของเคอร์เซอร์ไปที่ (2, 10)

```
Write (#27,'[',10,';',2,'H')
Write (#27,'[10;2H')
Write (#27,'[10;',2,'H')
Write (#27,'[',10,';2H)
```

โพรซีเดอร์ที่ทำงานเกี่ยวกับเคอร์เซอร์ ซึ่งสร้างจากรหัส ANSI ในรูปที่ 2 ได้แสดงไว้ในรูปที่ 3 อย่างไรก็ตาม โพรซีเดอร์ CursorXY ซึ่งทำหน้าที่เปลี่ยนตำแหน่งของเคอร์เซอร์ และโพรซีเดอร์อื่น ๆ ที่แสดงในรูปที่ 3 จะรับตัวแปรเป็นจำนวนเต็มทั้งหมดเพื่อความสะดวกในการเรียกใช้ การเรียกใช้โพรซีเดอร์ SaveCursorXY และ RestoreCursorXY ไม่ต้องส่งพารามิเตอร์ เมื่อเรียกใช้โพรซีเดอร์ SaveCursorXY ตำแหน่งของเคอร์เซอร์จะถูกเก็บไว้ เราสามารถเรียกใช้โพรซีเดอร์ SaveCursorXY ได้หลายครั้ง แต่ตำแหน่งของเคอร์เซอร์ที่เก็บไว้จะเป็นตำแหน่งครั้งหลังสุดที่เรียกใช้โพรซีเดอร์ SaveCursorXY และเมื่อต้องการเปลี่ยนตำแหน่งเคอร์เซอร์มายังตำแหน่งของเคอร์เซอร์ที่เก็บด้วยโพรซีเดอร์ SaveCursorXY ก็ให้ใช้โพรซีเดอร์ RestoreCursorXY

รหัสชุดต่อมาทำหน้าที่เกี่ยวข้องกับการลบอักษรบนจอภาพ รหัสชุดนี้

ชื่อรหัส	รหัส	คำอธิบาย
Cursor Position (CUP)	Esc[row;colH Esc[row;colf	เซตตำแหน่งของเคอร์เซอร์ที่ (col,row) มีสองรหัส จะใช้แบบไหนก็ได้
Cursor Up (CUU)	Esc[nrowA	เลื่อนเคอร์เซอร์ขึ้นจำนวนเท่ากับ nrow
Cursor Down (CUD)	Esc[nrowB	เลื่อนเคอร์เซอร์ลงจำนวนเท่ากับ nrow
Cursor Forward	Esc[ncolC	เลื่อนเคอร์เซอร์ไปทางขวาเท่ากับ ncol
Cursor Backward (CUB)	Ese[neolD	เลื่อนเคอร์เซอร์ไปทางซ้ายเท่ากับ ncol
Save Cursor Position (SCP)	Esc[s	เก็บตำแหน่งที่อยู่ของเคอร์เซอร์ไว้สามารถนำกลับมาใช้ได้ด้วย RCP
Restore Cursor Position (RCP)	Esc[u	เปลี่ยนตำแหน่งเคอร์เซอร์ไปยังตำแหน่งที่เก็บไว้ด้วย SCP

รูปที่ 2 รหัส ANSI ของเคอร์เซอร์

รูปที่ 3

```
 1: PROGRAM ANSI;
 2:
 3: Procedure CursorXY(col,row:integer);
 4: begin
 5:    Write(#27,'[',row,';',col,'H');
 6: end;
 7:
 8: Procedure CursorUp(nrow:integer);
 9: begin
10:    Write(#27,'[',nrow,'A');
11: end;
12:
13: Procedure CursorDown(nrow:integer);
14: begin
15:    Write(#27,'[',nrow,'B');
16: end;
17:
18: Procedure CursorRight(ncol:integer);
19: begin
```

รูปที่ 3 ต่อ

```
20:    Write(#27,'[',ncol,'C');
21: end;
22:
23: Procedure CursorLeft(ncol:integer);
24: begin
25:    Write(#27,'[',ncol,'D');
26: end;
27:
28: Procedure SaveCursorXY;
29: begin
30:    Write(#27,'[s');
31: end;
32:
33: Procedure RestoreCursorXY;
34: begin
35:    Write(#27,'[u');
36: end;
37:
```

ชื่อ	รหัส	คำอธิบาย
Erase Display (ED)	Esc[2J	ลบจอภาพทั้งจอแล้วให้เคอร์เซอร์ไปอยู่ที่ตำแหน่ง (1,1)
Erase Line (EL)	Esc[K	ลบบรรทัดตั้งแต่ตำแหน่งของเคอร์เซอร์ไปยังตำแหน่งท้ายสุดของบรรทัด

รูปที่ 4 รหัส ANSI ของการลบจอภาพ

```
38: Procedure ClearScreen;
39: begin
40:    Write(#27,'[2J');
41: end;
42:
43: Procedure ClearEOL;
44: begin
45:    Write(#27,'[K');
46: end;
47:
```

รูปที่ 5

แสดงในรูปที่ 4 มีด้วยกัน 2 รหัส คือ การลบจอภาพทั้งจอและการลบจอภาพจากตำแหน่งเคอร์เซอร์ไปยังท้ายสุดของบรรทัด โดยสร้างโพรซีเดอร์ ClearScreen และ ClearEOL ตามรหัส ANSI ดังรูปที่ 5

สำหรับการสร้างโพรซีเดอร์ในรูปที่ 3 และรูปที่ 5 เราสร้างโพรซีเดอร์หนึ่งโพรซีเดอร์จากรหัส ANSI หนึ่งรหัส เนื่องจากการทำงานของรหัส ANSI สามารถนำมาเทียบเท่ากับการทำงานของโพรซีเดอร์ที่สร้างได้ แต่สำหรับรหัส ANSI ที่แสดงในตารางในรูปที่ 6 จะมีความยุ่งยากขึ้น เนื่องจากรหัส SGR รหัสเดียวสามารถทำหน้าที่ทั้งเปลี่ยนแอตทริบิวต์ เปลี่ยนสีตัวอักษร และเปลี่ยนสีของแบคกราวนด์ได้ ตามค่าของพารามิเตอร์ mode ที่แสดงในรูปที่ 6 ดังนั้นเราจึงสร้างโพรซีเดอร์ SGR ในรูปที่ 7 เพื่อเรียกใช้รหัส SGR ในรูปที่ 6 จากโพรซีเดอร์ SGR เราจะสร้างโพรซีเดอร์ต่าง ๆ มาเรียกใช้โพรซีเดอร์ SGR ซึ่งแสดงในรูปที่ 8 โพรซีเดอร์ต่าง ๆ มีดังนี้

โพรซีเดอร์ SetAttribute จะเรียกใช้ SGR ในช่วงค่าโหมดตั้งแต่ 0 ถึง 8 โดยกำหนดค่าคงที่ไว้ใช้เรียกแทนตัวเลข

โพรซีเดอร์ ForeGround จะเรียกใช้ SGR ในช่วงค่าโหมดตั้งแต่ 30 ถึง 37 ซึ่งเป็นการเปลี่ยนสีของตัวอักษร โดยกำหนดค่าคงที่สำหรับแต่ละสีไว้ให้เรียกใช้ผ่านมาทางตัวแปร color โดยกำหนดหมายเลขของสีเริ่มจาก 0 เมื่อเรียกใช้โพรซีเดอร์ ForeGround แล้ว โพรซีเดอร์ ForeGround จะบวกค่าของ color เพิ่มขึ้นให้เริ่มต้นที่ค่า 30 ซึ่งเป็นช่วงของสีตัวอักษร แล้วเรียกโพรซีเดอร์ SGR ด้วยค่าดังกล่าว

โพรซีเดอร์ BackGround จะเรียกใช้ SGR ในช่วงค่าโหมดตั้งแต่ 40 ถึง 47 ซึ่งเป็นการเปลี่ยนสีของแบคกราวนด์ โดยกำหนดให้ใช้ค่าคงที่สำหรับ

ชื่อ	รหัส	คำอธิบาย
Set Graphics Rendition	Esc[modem	ทำงานตามค่าของ mode ที่ส่งมา
(SGR)	mode =	แอตทริบิ้วต์
	0	ยกเลิกคำสั่งที่เซตด้วย SGR ทั้งหมด
	1	ใช้ตัวอักษรเข้ม
	4	ใช้ตัวอักษรแบบขีดเส้นใต้ ใช้ได้เฉพาะจอโมโนโครมเท่านั้น
	5	ใช้ตัวอักษรแบบกระพริบ
	7	ใช้สีแบบ inverse
	8	ใช้ตัวอักษรแบบซ่อน คืออักษรสีดำบนพื้นสีดำ
	mode=	สีตัวอักษร
	30	สีดำ
	31	สีแดง
	32	สีเขียว
	33	สีเหลือง
	34	น้ำเงิน
	35	สีม่วง
	36	สีฟ้า
	37	สีขาว
	mode=	สีพื้นของตัวอักษร
	40	สีดำ
	41	สีแดง
	42	สีเขียว
	43	สีเหลือง
	44	น้ำเงิน
	45	สีม่วง
	46	สีฟ้า
	47	สีขาว

รูปที่ 6 รหัส ANSI สำหรับการเซตลักษณะการแสดงผล

```
48: Procedure SGR(mode:integer);
49: begin
50:     Write(#27,'[',mode,'m');
51: end;
52:
```

ปที่ 7

รูปที่ 8

```
53: CONST   Off = 0;     Intensity = 1;     Underline = 4;
54:         Blink = 5;     Inverse = 7;       Hidden = 8;
55:
56: Procedure SetAttribute(mode:integer);
57: begin
58:     SGR(mode);
59: end;
60:
61: CONST   Black = 0;       Red = 1;    Green = 2;    Yellow = 3;
62:          Blue = 4;    Magenta = 5;    Cyan = 6;     White = 7;
63:                                                          ➔
```

รูปที่ 8 ต่อ

```
64: Procedure ForeGround(color:integer);
65: begin
66:     SGR(30+color);
67: end;
68:
69: Procedure BackGround(color:integer);
70: begin
71:     SGR(40+color);
72: end;
73:
```

แต่ละสีเช่นเดียวกับโพรซีเดอร์ Foreground แต่ใช้ค่าบวกเพิ่มจาก color ที่ส่งมาต่างกัน

สำหรับรูปที่ 9 เป็นรหัสในการกำหนดการทำงานของโหมด wrap เมื่อให้การทำงานอยู่ในโหมด wrap หมายความว่าถ้าหากเมื่อใดที่ตำแหน่งเคอร์เซอร์เกินขนาดของหน้าจอ ไม่ว่าจะเกิดจากคำสั่ง Write หรือการป้อนข้อมูลยาว ๆ ผ่านคำสั่ง Read ตำแหน่งของเคอร์เซอร์จะเลื่อนมาที่บรรทัดถัดมา ส่วนการยกเลิกการทำงานในโหมด wrap จะทำให้เคอร์เซอร์ไม่เลื่อนมาที่บรรทัดถัดมาเมื่อตำแหน่งของเคอร์เซอร์เกินขนาดของหน้าจอ ดังนั้นเคอร์เซอร์จะอยู่ที่ตำแหน่งท้ายสุดของบรรทัด ในรูปที่ 10 โพรซีเดอร์ SetWrapMode จะหน้าที่เซตหรือยกเลิกการทำงานในโหมด wrap ขึ้นอยู่กับตัวแปร mode ว่ามีค่า True หรือ False ตามลำดับ

ตารางรหัส ANSI ในรูปที่ 11 แสดงการเปลี่ยนโหมดของจอภาพในโหมดต่าง ๆ การที่จะเปลี่ยนเป็นโหมดใด ๆ ได้นั้น จะต้องใช้จอภาพและการ์ดที่สนับสนุนการใช้งานจอภาพในโหมดนั้นด้วย นอกจากนั้นสำหรับการเปลี่ยนจอในโหมด 14 ถึง 19 จะต้องใช้ดอสเวอร์ชัน 4.0 เป็นต้นไป โพรซีเดอร์

รหัส	คำอธิบาย
Esc[?7h	เซตโหมดการทำงานในโหมด wrap
Esc[?7l	ยกเลิกการทำงานในโหมด wrap

รูปที่ 9

```
74: Procedure SetWrapMode(mode:boolean);
75: begin
76:     if mode then
77:         Write(#27,'[?7h')
78:     else
79:         Write(#27,'[?7l');
80: end;
81:
```

รูปที่ 10

VideoMode ในรูปที่ 12 จะใช้ค่าคงที่ BW40, CO40, BW80 และ CO80 แทนค่าคงที่สำหรับโหมด 0 ถึงโหมด 3 ตามลำดับซึ่งเป็นเท็กซ์โหมด ส่วนโหมดอื่นเป็นกราฟิกโหมดไม่ได้กำหนดค่าคงที่ไว้ ให้ใช้ค่าตัวเลขส่งไปเรียกโพรซีเดอร์ VideoMode ได้เลย

เมื่อผู้อ่านเปลี่ยนโหมดของจอภาพเป็นกราฟิกโหมดแล้ว จะสามารถใช้คำสั่งต่าง ๆ ที่แสดงผลในเท็กซ์โหมดได้ อย่างเช่น WriteLn, ReadLn นี่เป็นคุณสมบัติประการหนึ่งของรหัส ANSI

ชื่อ	รหัส	คำอธิบาย
Set Mode	Esc[modeh	เปลี่ยนโหมดของจอภาพตาม mode ที่ส่งมา
(SM)	mode =	
	0	40×25 จอขาวดำ
	1	40×25 จอสี
	2	80×25 จอขาวดำ
	3	80×25 จอสี
	4	320×200 จอสี 4 สี
	5	320×200 จอโมโนโครม
	6	640×200 จอโมโนโครม
* EGA และใช้ดอส 4.0	14	640×200 จอสี
* EGA และใช้ดอส 4.0	15	640×350 จอโมโนโครม
* EGA และใช้ดอส 4.0	16	640×350 จอสี 16 สี
* EGA และใช้ดอส 4.0	17	640×480 จอสี 2 สี
* EGA และใช้ดอส 4.0	18	640×480 จอสี 16 สี
* EGA และใช้ดอส 4.0	19	320×200 จอสี 256 สี

รูปที่ 11 รหัสเปลี่ยนโหมดจอภาพ

```
82: CONST   BW40 = 0;   CO40 = 1;   BW80 = 2;   CO80 = 3;
83:
84: Procedure VideoMode(mode:integer);
85: begin
86:    Write($27,'[=',mode,'h');
87: end;
88:
```

รูปที่ 12

```
89: CONST     Vseg = $A000;   OverPut = 0;   XorPut = $18;
90:          AndPut = $08;        OrPut = $10;
91:
92: Procedure GraphPort(index,data:integer);
93: begin
94:    Port[$03CE]:=index;
95:    Port[$03CF]:=data;
96: end;
97:
98: Procedure PlotXY(x,y,color,puttype:integer);
99: var   offset,dummy : integer;
100:       mask         : byte;
101: begin
102:    offset:=y*80+(x div 8);
103:    mask:=$80 shr (x mod 8);
104:    dummy:=mem[Vseg:offset];
105:    GraphPort(0,color);
106:    GraphPort(1,$0F);
107:    GraphPort(3,puttype);
108:    GraphPort(8,mask);
109:    mem[Vseg:offset]:=1;
110:    GraphPort(0,0);
111:    GraphPort(1,0);
112:    GraphPort(3,0);
113:    GraphPort(8,$FF);
114: end;
115:
```

รูปที่ 13

โพรซีเดอร์ต่าง ๆ ที่เราสร้างขึ้นแต่ต้น เช่น ClearScreen, CursorXY หรือโพรซีเดอร์อื่น ๆ ก็ยังสามารถใช้ได้ ดังนั้นผู้อ่านจึงสามารถควบคุมการแสดงผลในโหมดกราฟิกได้ทันที นอกจากการเขียนรูปซึ่งรหัส ANSI ไม่ได้สนับสนุนไว้ เนื่องจากรหัส ANSI กำหนดขึ้นเพื่อให้สามารถใช้งานได้พอสมควรเท่านั้น ซึ่งก็เพียงพอแก่การเขียนโปรแกรมที่ไม่ซับซ้อน แต่ถ้าผู้ใช้ต้องการวาดรูป เช่น เส้นตรง, วงกลม โดยใช้การเปลี่ยนโหมดจอภาพของ ANSI ก็สามารถสร้างโพรซีเดอร์ที่ทำหน้าที่ดังกล่าวขึ้นได้ โดยใช้การกระทำบนหน่วยความจำของจอภาพโดยตรง ซึ่งต้องขึ้นอยู่กับประเภทของจอภาพที่ใช้

สำหรับกราฟิกโหมดที่นิยมในปัจจุบันคือ โหมด 16 ซึ่งทำงานบนจออีจีเอหรือวีจีเอ ดังนั้นจึงขอยกตัวอย่างการพล็อตจุดบนหน้าจอสำหรับการทำงานในโหมดนี้ ดังแสดงในรูปที่ 13 ให้ชื่อว่าโพรซีเดอร์ PlotXY ค่าของตัวแปร color ซึ่งเป็นสีที่จะใช้พล็อตมีลักษณะต่างจากค่าสีของ ANSI ค่าของตัวแปร color สำหรับเรียกใช้โพรซีเดอร์ PlotXY มีค่าดังนี้

0 = สีดำ
1 = สีน้ำเงิน
2 = สีเขียว
3 = สีฟ้า
4 = สีแดง
5 = สีม่วง
6 = สีน้ำตาล
7 = สีเทาเข้ม
8 = สีเทา
9 = สีน้ำเงินเข้ม
10 = สีเขียวเข้ม
11 = สีฟ้าเข้ม
12 = สีแดงเข้ม
13 = สีม่วงเข้ม
14 = สีเหลือง
15 = สีขาว

รหัส	คำอธิบาย
Esc[key;text;p	กำหนดให้คีย์ที่มีค่า key ถูกแทนเป็น text

รูปที่ 14 รหัส ANSI สำหรับกำหนดคีย์บอร์ด

สำหรับการเขียนโปรแกรมในโหมดกราฟิกบนจออีจีเอจะไม่ขอกล่าวถึง ผู้อ่านสามารถศึกษาเพิ่มเติมจาก PROGRAMMER'S GUIDE TO THE EGA/VGA เขียนโดย Sutty and Blair

รหัส ANSI ชุดสุดท้ายเป็นการทำงานเกี่ยวกับคีย์บอร์ด โดยอนุญาตให้เปลี่ยนค่าของคีย์บอร์ดได้ เช่น ถ้าเราต้องการให้กดอักษร A แล้วได้ค่าของอักษรอื่น หรือให้กดอักษร A แล้วได้ค่าของสตริงที่แสดงข้อความก็ได้ รูปที่ 14 แสดงรหัส ANSI ที่ใช้ในการเปลี่ยนค่าของคีย์บอร์ด ส่วนตารางในรูปที่ 15 เป็นค่ารหัสของคีย์บอร์ดที่ใช้ส่งค่าดังกล่าว สำหรับโพรซีเดอร์ AssignKey ในรูปที่ 16 จะรับค่าคีย์เป็นสตริงเพื่อให้สามารถรับค่าของ extended key ได้ และพารามิเตอร์ text จะใช้เครื่องหมาย "..." เป็นตัวกำหนดข้อมูลที่จะส่ง ตัวอย่างต่อไปนี้จะเปลี่ยนค่าคีย์ของอักษร A เป็นอักษร B โดยเรียกใช้โพรซีเดอร์ AssignKey และใช้ค่าของคีย์ตามตารางในรูปที่ 15 ซึ่งแสดงค่าคีย์ของ A=65 ของ, B = 66, F1 = 0;59, F2 = 0;60 และ F3 = 0;61

```
AssignKey ('65','66')
```

หรือเปลี่ยนคีย์ F1 เป็นข้อความ "This is F1" ที่ตามด้วยคีย์ ENTER

```
AssignKey('0;59',' "This is F1";13')
```

เมื่อเราทำงานอยู่ในดอสคีย์ F3 หมายถึงให้นำคำสั่งเก่ากลับมา แต่จะให้ผู้ใช้กดคีย์ ENTER อีกครั้งเพื่อเป็น

Key	Code			
	Alone	Shift-	Ctrl-	Alt-
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0.96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
Home	0;71	55	0;119	---
Up Arrow	0;72	56	---	---
Pg up	0;73	57	0;132	---
Left Arrow	0;75	52	0;115	---
Down Arrow	0;77	54	0;116	---
End	0;79	49	0;117	---
Pg Dn	0;81	51	0;118	---
Ins	0;82	48	---	---
Del	0;83	46	---	---
PrtSc	---	---	0;114	---
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0.47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33	---	0;120
2	50	64	---	0;121
3	51	35	---	0;122
4	52	36	---	0;123
5	53	37	---	0;124
6	54	94	---	0;125
7	55	38	---	0;126
8	56	42	---	0;127
9	57	40	---	0;128
0	48	41	---	0;129
-	45	95	---	1;130
=	61	43	---	0;131
Tab	9	0;15	---	---
Null	0;3	---	---	---

รูปที่ 15

การยืนยัน เราจะกำหนดให้คีย์ F2 ทำหน้าที่เหมือนคีย์ F3 ตามด้วย ENTER โดยใช้โพรซีเดอร์ AssignKey กำหนดดังนี้

```
AssignKey('0;60','0;61;13');
```

รูปที่ 17 แสดงโพรซีเดอร์ TextDemo ซึ่งแสดงตัวอย่างในการเรียกใช้โพรซีเดอร์ต่าง ๆ ในเท็กซ์โหมดเพื่อรับข้อมูลแบบสตริง 5 บรรทัด เพื่อเป็นตัวอย่างในการทำงาน และรูปที่ 18 แสดงโพรซีเดอร์ GraphicsDemo แสดงตัวอย่างการทำงานในโหมดกราฟิก ซึ่งแสดงแถบของสีต่าง ๆ 15 แถบสี (ไม่รวมแถบสีดำ) ในส่วนของโพรซีเดอร์ GraphicsDemo ต้องทำงานบนดอสตั้งแต่ 4.0 ขึ้นไป จะเห็นว่าส่วนท้ายสุดของโพรซีเดอร์ GraphicsDemo จะเซตสีตัวอักษรเป็นสีเขียวและจะจบโปรแกรมเมื่อกลับมาในส่วนของโปรแกรมหลัก ดังนั้นเมื่อกลับมาที่ดอสผู้ใช้จะเห็นว่าสีของตัวอักษรในดอสจะเป็นสีเขียวเนื่องจากการใช้รหัส ANSI ในการเซตสีตัวอักษรจะมีผลถึงการทำงานในระดับดอสด้วย รวมทั้งการเซตโหมดการทำงานในโหมด wrap ก็จะยังคงมีผลอยู่เช่นกัน หวังว่าผู้อ่านคงได้รับประโยชน์จากยูทิลิตี้ที่สร้างขึ้นมาทั้งหมดนี้บ้างตามสมควร

รูปที่ 16

```
116: Procedure AssignKey(key,text:string);
117: begin
118:    Write(#27,'[',key,';',text,'p');
119: end;
120:
```

รูปที่ 17

```
121: Procedure TextDemo;
122: const  item : array[1..5] of string =
123:       ( 'NAME   :',
124:         'AGE    :',
125:         'SEX    :',
126:         'STATUS :',
127:         'SALARY :'  );
128: var i : integer;
129:     input : array[1..5] of string;
130: begin
131:    VideoMode(C040);
132:    SetAttribute(Off);
133:    BackGround(Black);
134:    ForeGRound(Green);
135:    ClearScreen;
136:    for i:=1 to 5 do
137:       WriteLn(item[i]);
138:    SetAttribute(intensity);
139:    ForeGround(Yellow);
140:    SetWrapMode(False);
141:    for i:=1 to 5 do begin
142:       BackGround(White);
143:       CursorXY(9,i);
144:       SaveCursorXY;
145:       ClearEOL;
146:       RestoreCursorXY;
147:       ReadLn(input[i]);
148:       input[i]:=Copy(input[i],1,30);
149:       BackGround(Black);
150:       RestoreCursorXY;
151:       Write(input[i]);
152:       ClearEOL;
153:    end;
154:    CursorXY(10,24);
155:    Write('ENTER for graphics demo');
156:    ReadLN;
157: end;
158:
```

รูปที่ 18

```
159: Procedure GraphicsDemo;
160: var  x,y : integer;
161: begin
162:    VideoMode(16);
163:    CursorXY(1,24);
164:    Write('COLOR:');
165:    for y:=0 to 159 do begin
166:       CursorXY(7,24);
167:       Write(y div 10);
168:       for x:=0 to 639 do
169:          PlotXY(x,y,y div 10,OrPut);
170:    end;
171:    CursorXY(33,24);
172:    SetAttribute(Off);
173:    ForeGround(Green);
174:    Write('Press ENTER');
175:    ReadLn;
176:    VideoMode(C080);
177: end;
178:
179: BEGIN
180:    TextDemo;
181:    GraphicsDemo;
182: END.
```